NEA Computer Science

1 Analysis:

1. Background and identification of problem:
   1. Background:

Caistor Grammar School, (CGS), is a secondary school and sixth form college which teaches a variety of subjects at GCSE and A-level including further maths.

* 1. Identification of problem:

The further maths department has been doing the A-level AQA course for many years. The teachers have been looking for an interactive resource in which to help teach network flows to students within the discrete section of both AS and A-level content, utilising a visual representation to convey the topic through the direct use of graphs. This section of the course primarily requires students to find the maximum flow when given maximum capacities and potential minimum capacities of each edge, acting as a lower bound as well as a minimum cut. This acts as the upper bound which is based on a cut value found once it has been drawn, leading to the solution regarding the maximum flow- minimum cut theorem, (the theorem solved by network flows). To demonstrate this, the ability to orientate the graph based on the pre-defined layout, would reduce confusion when teaching rather than nodes being in different positions in relation to other nodes. While a step-by-step approach would be beneficial to demonstrate the process and notation for each problem. Due to the aim of the application being its interactivity, this enables the teacher to target specific questions or sub-parts of this topic that are often missing from the textbook such as a maximum node capacity.

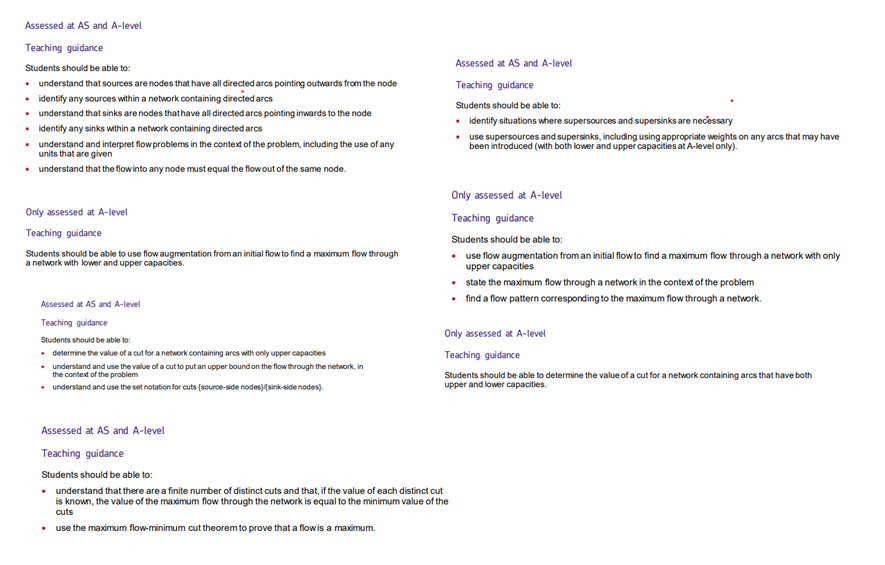
And yet, even though this could be taught using PowerPoints and textbooks, it would be more engaging and effective if the questions were focused on a sub-topic, broadening the potential question set. This makes solutions more accessible, while acting more versatile to the teacher’s lesson plans. In the case of this application, a more engaging method could be for the user to be able to find their own minimum cuts using the graph which was inputted instead of an algorithm drawing it, as well as use the changeability of the graph to discover the possible effects of minimum edge capacities, specifically affecting their solution. This could also be useful to students not just in class time but independently in regard to understanding the topic as a whole, reinforcing their learning as a whole as well as in specific areas.

1. Description of current resources:

Currently, PowerPoints are used as the primary teaching tool, usually covering the material in 1 or 2 PowerPoint files. Although this allows the ability once projected to create a step-by-step solution through annotations on the whiteboard, this often leads to a multitude of the slides being ignored, often due to the diagram being redrawn or due to discussions and work already annotated before arriving at the slide. These resources however covers most of the sub-topics, acting as the given exemplar for the lesson’s agenda, focusing on class interaction to initially understand the topic. These PowerPoints can be accessed outside class time as a resource to enable further reinforcement of knowledge. Furthermore, between having 2 textbooks, (both the Kerboodle textbook which separates As and A-level into separate books as well as a discrete only textbook), which are either in the student’s possession or distributed as necessary, this has several exercises and explanations to further develop student’s knowledge. While these are physical books, they can also be accessed online allowing students to have the additional opportunity to practice in their own time.

1. Research:

Network flows is used to determine the maximum flow – minimum cut theorem, ensuring the optimisation regarding the maximum flow within a graph. The current specification is shown in the image below, depicting the various sub-topics within the problem which are further explained below.



Specification:

Network flows is often represented by directed, connected graphs with 6 main components, as follows:

* Node/ vertex – a point on the graph,usually represented by a circle with name/ identifier in the centre as shown in the left image below.
* An edge which connects two nodes together, represented by an arrow as depicted in the right image below, which in this case is always uni-directional and weighted. The weight refers to the maximum flow you can send down the edge often due to physical constraints.
* The graph is connected meaning there is always a path from any one node to another with no one node being isolated from the others.
* The graph is planar, meaning that the graph can be drawn without any edges crossing as this enables an accurate value of a cut, (which is explained in step 4 below).
* A source – A node in which only has edges going from itself to another node, with no incoming edges, usually with a node identifier of S. This is the start point of the graph.
* A picture containing text, handwriting, ink, font

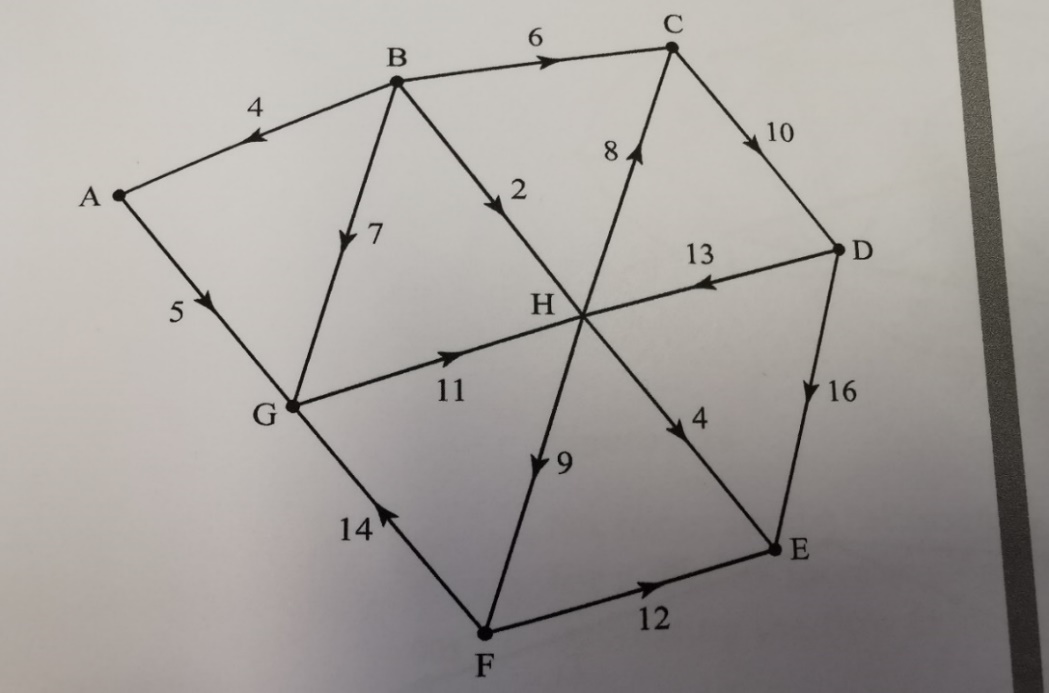
  Description automatically generatedA picture containing text, handwriting, ink, font

  Description automatically generatedA sink – a node in which only has edges going to itself with no outgoing edges, usually with a node identifier of T. This is the end point of the graph.

Arrow head = direction

At its core, network flow is a topic used to solve the maximum flow/minimum cut theorem, with each weight representing a maximum capacity to be sent down an edge. This is represented by the number adjacent to each edge and directly correlates to the flow you can send down, acting as an upper bound, with the lower bound for each edge being 0 unless specified due to a minimum capacity, (which is explained within the last bullet point of ‘Other factors that affect the problem’). Below is a step by step walk though of this type of problem, with nodes referenced relating to example 1 if not stated.

The problem below is an example of a problem from one of the current resources for network flows within “Further Mathematics Discrete” by Nick Greene.



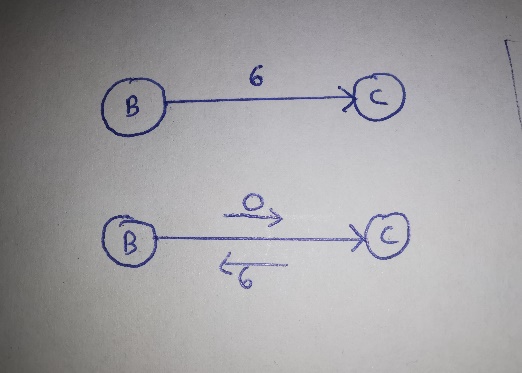
**Example 1:**

Step 1: Finding the source and sink:

Although a source is usually the left-most node and the sink is the right-most node, as shown in the image above this is not always the case. To determine which nodes are a source and sink, we can assess each node individually to see if a node only has edges which are outgoing, ingoing or a combination of both. For example, node A has an outgoing edge to the node G as well as an ingoing edge from B, therefore meaning G is neither a source or sink. On the other hand, node B has 4 outgoing edges, (to nodes A, C, G and H), but no incoming edges, resulting in it being a source. Node E can be identified as a sink by the same process due to the node having 3 incoming edges, which were from nodes D, F and H.

Step 2: Setting the problem up:

For every edge, there is always a forwards and backwards flow, where the forward flow follows the direction indicated by the arrow on the edge and the backward flow is in the opposite direction. The total sum of these flows is equal to the maximum capacity with neither being negative. Therefore, initially set the backward flow of one edge equal to the maximum capacity of that edge and the forward flow to be 0 as there is no initial flow within the graph. This is repeated until there are no more edges left that haven’t been visited. This is shown for a maximum capacity of 6 in the image below, which is the edge between B and C in example 1.



Before :

After:

Step 3: Finding the maximum flow:

A maximum flow is the sum of all the flows in each path, with a path being a sequence of vertices and edges of a graph which are not repeated going from the source to the sink.

A picture containing text, drawing, sketch, handwriting

Description automatically generatedA graph paper with writing on it

Description automatically generatedFor example, BAGHE has a flow of 4, due to each weight acting as a limiting capacity, with maximum flow possible being the remaining forward flow that can gained for the all the edges on the path. As BA and HE has a maximum capacity of 4 whereas AG and GH have a greater capacity of 5 and 11 respectively. This means for each edge within this path, the forward flow is 4 and the backward flow = original backward flow – forward flow. This means for an edge such as BA this will have a backward flow of 0, whereas AG will have a backward flow of 1. Keep augmenting a flow until no more flow can go from source to sink. The image on the right depicts a usual format to represent your augmented paths, with the path being on the left of the table and flow on the right. The image on the left shows the flow of that path, without the arrows and weights of the other edges not in the path for simplification.

However, if a further path of BGHCDE is made, the limiting capacity would be BG and GH as a maximum of 7 can be sent. Although GH does have a maximum capacity of 11, a forward flow of 4 is already obtained through the first path, therefore 11- 4 is 7.

A graph paper with arrows and lines

Description automatically generatedA graph paper with writing on it

Description automatically generatedIn the case of example 1, the maximum flow is 19. This provides a lower bound as there is always a possibility that a greater flow exists by being more optimised. Below is a potential maximum flow solution, however from the same values, there is several possible solutions with varying paths taken.

Step 4: Finding the minimum cut:

A minimum cut is when a line is drawn through the graph splitting it into two sections with the source and sink being in opposite sections, measuring the maximum flow through the edges which are intersected. The source side is on the left hand side of the cut notation and the sink side is on the right, ({source side nodes}/{sink side nodes}.

A graph paper with a drawing of a diagram

Description automatically generatedIf an edge, (dictated by the arrow on the edge itself) is going from the source side to sink side, the forward flow is added to the minimum cut. However, if the edge is going from the sink side to the source side, this is ignored and doesn’t affect the minimum cut if the graph only has maximum capacities.

For the cut in the image above, (C1), the given notation would be {B, C}/{A,D,E,F,G, H}, where the curly brackets depict which side each node falls. Therefore, the relevant edges are BA, BG, BH, HC and CD, evident by their intersection with the orange line. As HC goes from sink to source this is ignored, meaning the minimum cut = 4+7+2+10 = 23.

The optimum cut usually goes through the edges which has backward flow equal to 0 as no more flow can be sent so a limiting edge, (highlighted by purple circle in the image above) or edges which go from sink section to the source section. The optimised cut is shown by the cut C2with a value of 19. This provides an upper bound as this measures the capacity of the edges, of which may have edges that have a greater capacity than can be sent at one time.

Step 5: The maximum flow minimum cut theorem:

Through the maximum flow being the lower bound and the minimum cut being the upper bound, this creates a given range for the optimised flow. If these 2 values match, then the optimised flow is found. This allows a certainty that no more flow can be found to be sent through the network. If these don’t match, this means a further flow can be found or a better cut can be made. This is particularly useful for sewers as well as network filtering as ensures the systems meets demand, reducing unnecessary delays of getting to the sink.

In example 1, this is 19 as both the upper bound and lower bound is the same.

Other factors of the problem which are optional:

* Supersources also occur when there are multiple sources, resulting in a node connecting the sources together at a single point, with edges containing a minimum weight of each edge connected from the source. This often causes the nodes to be renamed S1 and S2. This is done in between step 1 and 2.
* A graph paper with a diagram

  Description automatically generatedSupersinks are similar to supersources, however this is found when there are multiple sinks instead of sources with edges connecting from the sink to the supersink being of a minimum weight in relation to the edges connected from a node to a sink. This often causes the sinks to be renamed T1 and T2. This is also done in between step 1 and 2.

A diagram of a diagram on a graph paper

Description automatically generated

* A paper with writing on it

  Description automatically generatedNode capacities occur when rather than an edge having a maximum capacity, the node itself has one. This is often represented by the node being split into two, with the first node usually having the identifier name followed by 1, with the other being followed by 2. These two nodes are connected by an edge acting as a limiter to the possible flow where the node capacity will become the maximum capacity of the new edge. This causes the ingoing flow to be connected to the node followed by 1 and the outgoing flow to be connected to the node followed by 2 for the original node. This is done in between step 2 and 3 and seen in the image below for the node B with a node capacity of 25.
* Network flows can also have minimum capacities, where a minimum amount of flow is required to flow through certain edges. This could be to reduce wear on other edges/pipes or prevent an edge going dry. This is often represented by 2 values along the edge separated by commas, with the first representing the minimum capacity and the latter being the maximum capacity, as shown in the image below.

A paper with writing on it

Description automatically generatedh

In between steps 2 and 3, starting at the sink and working backwards to the source, take each node individually and compare minimum flows of both incoming and outgoing edges. If they are equal, the forward flow of the edges will equal the minimum capacities. If the incoming flow is greater than the outgoing flow, increase the necessary flow of the outgoing edges to match, almost increasing the minimum capacity of that edge. If the outgoing flow is greater than the ingoing flow, increase the necessary flow of the ingoing edges to match, almost increasing the minimum capacity of those edge. This can be based on the future nodes, ensuring minimum flow is the smallest value found while still satisfying all minimum capacities for each edge. Combine the minimum flow with step 3, meaning total maximum flow is minimum flow + flow found in step 3. This ensures all minimum capacities are met as shown by the 3 stages in the image above.

A paper with writing on it

Description automatically generatedFor minimum cuts within step 4, if an edge goes from sink to source, instead of ignoring, minus the minimum capacity of that edge from the minimum cut found as flow moving backwards within the graph. This is shown within C2 within the image below for edges BC and DC, evident by them being subtracted.

1. Identification of the Prospective User(s):

Further Maths teachers would be the primary user, aiding their teaching resources as interactive way to utilise examples for both the AS and A level syllabus. This means this should be easy to use and understand, highlighting key aspects or patterns within problems. Students could also act as a secondary user, aiding their current knowledge, using it as a further resource for questions and revision, allowing focus on particular sub areas of the topic.

1. Identification of User Needs:
   1. The students should be able to add to the graph directly, adding each node and edge with necessary parameters – this could include the ability to delete node and/or edges and have an optional minimum capacity.
   2. The students should be able to add an augmented flow into the graph with necessary parameters, updating forward and backward flows based on inputs.
   3. The ability to determine which nodes are sources and sinks – it could add a supersource or supersink automatically onto the screen if multiple sources and/or sinks with the ability to hide this if need.
   4. The program should be able to directly draw the nodes and edges on the graph based on the user coordinates inputted, allowing proper orientation of graph to match existing versions.
   5. The program should be able to have the ability to create node capacities, displaying the node in two parts with a connecting edge, based on the direction in which the edges have come from.
   6. The ability to work out the maximum flow when given the maximum capacities – it could show the path and flow, showing the process of the theorem with the ability to pause at various points within the maximum flow to allow teaching points to be made.
   7. The program should also be able to solve minimum capacities, checking it is possible, (if minimum capacities <= maximum capacities out of a given node).
   8. The ability to create an optimum minimum cut, where the user can add their own. It would be good to accompany the notation accompanied with each cut as well as have the ability to hide them.
   9. A diagram of a diagram

      Description automatically generated with medium confidenceThe ability to point out key tips for the problem or patterns as the step by step solution progresses
   10. The ability to evaluate maximum flow and minimum cuts, found for the maximum flow – minimum cut theorem, finding the optimal flow.
2. Data Flow/ process diagram:

Data flow diagram:

This diagram shows the flow of data throughout the program based on the user’s views by considering the program in two parts, one loop enables the inputs for the node and edges to be inputted and displayed, whereas the other loop branches two parts, the max flow component and min cut component evaluating at the end with a message if the two values found are the same.

A diagram of a flowchart

Description automatically generatedProcess Diagram:

The shows the layout and order in which the user will be able to see each part and component they will be seen/ accessed, highlighting the core functionality seen step by step.

1. Objectives of the project:
   1. The program should have the ability to input and display the required information into/on the screen for all required parts of the network flow problem. This could be through an interface on the left side of the screen.

Specific Objectives:

1. Allow the user to add a node through a node identifier within a given length of characters, with coordinates for its position on the axis on the screen.
2. Allow the user to add edges with the mandatory inputs being the minimum and maximum capacities, the node it came from and the node it has gone too.
3. Allow the user to add optional minimum capacities with a given weight as well as allow the ability to check for any given node the minimum ingoing/outgoing capacity is less than the outgoing/ingoing total respectively.
4. Allow the user to add node capacities for given node identifier and weight as an optional input.
5. Allow the user to add flows into the graph created when given a valid path and flow.
6. Allow the user the ability to delete nodes/edges/flows that are entered incorrectly.
7. The ability to display the nodes and edges in a clear, coherent manner, grouped together based on a corresponding node.
8. The ability to display the augmented paths added within a table format in a clear and coherent manner
   1. Identifying possible sources and sinks, storing these for each given node.

Specific Objectives:

1. The ability to determine based on the incoming and outgoing edges, whether a node is a source or sink.
2. If there is multiple sources or sinks, it has the ability to create a new node for the supersource and/or supersink, with valid weight which has no impact on the network flow graph, (greater than outgoing edges of the sources or ingoing edges of sinks).
3. Ability to find the position of a given node and edge for a supersource/supersink
4. Display the weights on the edges, which is not amendable to the user.
5. Error should be identified if there isn’t at least both a source and sink.
6. Ability to hide supersources and/or supersinks corresponding node and edges using a button.
7. Ability to account for node capacities within the sources and sinks, shifting edges.
   1. Drawing the graph based on the user coordinates, adding node capacities if needed.

Specific Objectives:

1. Ability to display a coordinate axis onto the area to aid the user entering coordinates for the nodes.
2. Represent a node as a circle with the node identifier within.
3. Represent an edge as an arrow from one node to another with the capacity, with 2 smaller arrows, both parallel to the edge representing the forward/backward flow and with its corresponding flow values.
4. The ability to draw the graph based on the coordinates given by the user as seen on the screen by the grid like structure with supersource and supersinks added if valid positions found.
5. The ability to check if the new node or edge added is valid and does not intersect another node and edge.
6. The ability to only display minimum capacities within the graph if and only if there is at least 1 with a minimum capacity greater than 0.
7. For any given node capacity, it has the ability to split the node into 2, with a given edge in between, without colliding with other objects on the graph, remaining the node identifier to a suitable alternative.
8. Ability to segment the flows into individual steps so can be seen and paused if needed.
9. Ability to change any type of source or sink to a different colour/shape to make it more identifiable.
10. Ability to hide/show node capacities when drawn onto the graph.
    1. Finding maximum flow for within a given graph:

Specific Objectives:

1. Able to find all possible paths from a source to a sink. If minimum capacities occur, minimum flow should be found before additional flow can be found, accounting for the augmented paths added by the user.
2. Ability for user to preset the flow within the graph, changing the necessary flows within the edges.
3. Ability to account for added supersource/supersinks if the valid position can be found.
4. Ability to traverse node capacities within the graph, without exceeding the value given.
5. Ability to display path and there flows in a separate area on screen with the option to split this into flow for minimum capacities and additional flow with a total stated.
6. Ability to update the screen in regards to forward and backward flows for each node.
7. Ability to make edges with a raised minimum capacity more identifiable once it has changed.
8. Allows backward flow if 0 to be more identifiable for better visualisation of remaining paths left to augment or minimum cut once maximum flow is found.
9. Ability to do the algorithm in segments, adding the flow and paths one at a time
   1. Finding the minimum cut within a given graph, and be able to calculate a cut based on lines drawn by the user:

Specific Objectives:

1. The ability to allow the user to draw their own cut onto the graph, automatically labelling it if valid.
2. Ability to determine and analyse the cut drawn and find cut value, ignoring or subtracting the minimum capacities if going from sink 🡺 source.
3. Ability to account for supersource, supersinks and node capacities when calculating the cut value.
4. The ability to display a cut using the specific notation on the screen, based on the users cut.
5. Ability to display calculation to find the cut value.
6. Ability to hide the drawn cuts from the screen.
7. Ability to randomly generate colour of line to make it easier to identify.
   1. The maximum flow – minimum cut theorem is shown if proved.

Specific Objectives:

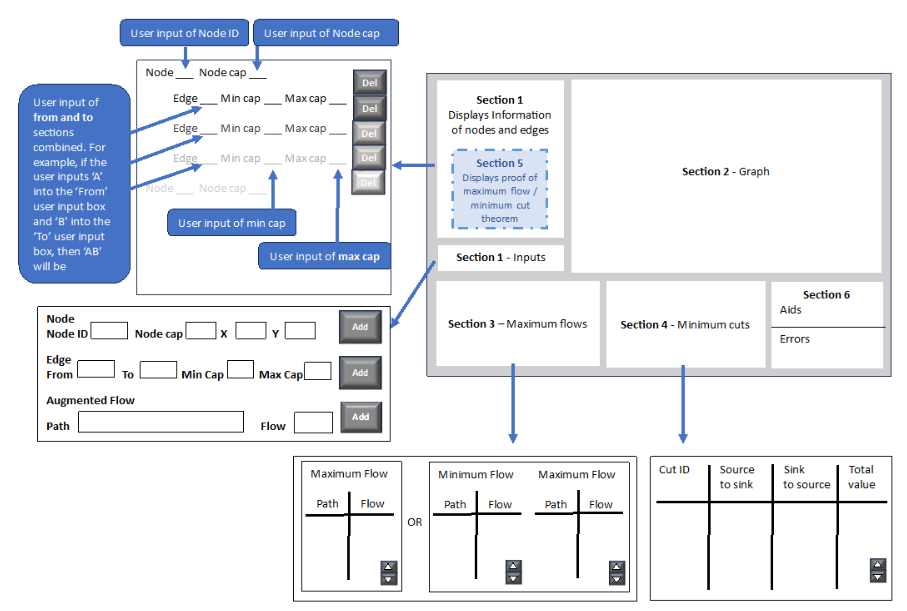
1. Compare the maximum flow and minimum cut, displaying an appropriate message to show optimum flow has been found.

I have expanded upon the user’s needs to develop the objectives grouping some together, hence the number of major points (6) is less than the number of user needs (10).

1. Acceptable limitations:

As discussed with the further maths department, due to this being a teaching tool, the number of nodes is on the smaller scale in relation to other graphs, with a limit of 15 nodes. This means the nodes will still be visible and the user will be able to follow the theorem without getting overwhelmed with a plethora of edges and nodes. The graph must also be planar so that edges can not to cross, therefore non planar graphs will not be able to be drawn. This enables tracking the maximum flow and minimum cut components to be easier. This also means visually each node will have a greater magnification, making it more interactable to a class which will have students at variable distances away. The final limitation is that the directed graph will have no cycles.

1. HCI

The screen will be split into multiple sections signifying varying parts of the problem as shown in the image below, with each section matching the colour scheme and style to be consistent.

Section 1

Initially, within section 1, the user will enter various parameters for either a node, edge or augmented flow within the graph into textboxes which are labelled. For the node, the user will enter a node identifier, maximum capacity, and x y coordinates. Whereas for the edge, the user will enter the node identifiers for where the edge is going to and from, the minimum capacity and the maximum capacity. For the augmented flow, a path is entered as well as the flow within that path. This means there will be 4 input boxes for entering information for a node or edge and 2 textboxes for augmented flow as depicted in the magnification of Inputs in this section on the diagram above. Due to the various requirements of the parameters for the nodes and edges, after the ‘Add’ button is pressed, the text for each textbox will be validated, ensuring correct data has been added. For example, ensuring the maximum capacity is an integer. If there is at least 1 textbook which has incorrect inputs, the whole node or edge will be ignored. This will then be displayed in the scrollbar above, with which allows the user to delete nodes (and the edges connecting to that node) as well individual edges. This allows the user to resubmit the nodes and edges they may have submitted wrong or even to adapt the graph to different parts of a question. Each edge will be displayed under the node from which it comes from, grouping those edges together as referenced in the magnification of section 1 in the image above. Furthermore, through the ability to scroll up and down, this allows the user to view all the data no matter whether it fits within the given area for displaying that information.

For this section, there will be 5 buttons, 3 ‘Add’ buttons allowing the ability to submit data for nodes and edges and paths, while the other 2 allow the scrollbar to move up and down.

Section 2

Within section 2, this is the area where the graph is drawn onto the screen as well as the area the user will draw on cuts. Before the user enters any information for nodes and edges, this area will be a coordinate grid which can act as a reference to the users when inputting the coordinates of the nodes. When the user is entering the nodes and edges, the nodes, edges with a forward flow and backward flow should automatically appear onto the grid at that location, allowing the graph to be built up as the user is entering the necessary information, with supersources, supersinks and node capacities automatically being added if appropriate.

The graph will also act as an interface for cuts to be drawn, where the user will be able to build each one up in segments by clicking the mouse within the area. However, once the space bar is pressed, the cut will be fixed in place and validated. For example, checking if all the sources are on one side and all the sinks are on the other side. Once validated, the cut will change to a random colour, allowing it to be distinct and easier to see all the segments, while it is analysed to find cut values as well as the cut in terms of set notation.

Moreover, patterns will be highlighted through the changing of colours or shape, enabling teaching points to be made or generally aid the understanding of the graph. For example, if the backward flow is 0, this would be highlighted as well as source and sinks will be different from normal nodes, signalling the start and end of the graph.

Along the top edge of this section, there are 4 buttons which have dual phases, with two sets of texts which are displayed one at a time. These will be displayed once the user has finished entering inputs for the graph and clicked the button, signalling for the graph to be sold. The first, either enables, the ability to solve the graph in regards to finding a maximum flow and drawing the cuts, signifying the creation of the graph is finished and the user does not have any more edges and nodes to add or delete, while the other enables the graph to be edited, removing all cuts which have already been drawn. The other 3 buttons allow the ability to show or hide 3 features, which are a supersource, node capacity and a supersink. This allows the user to emphasise the impact they can create to a graph.

Section 3

For section 3, this displays the paths and flow for the graph in a table format, utilising the scrollbar to display paths no matter whether they all fit in the given area. If any of the edges have a minimum capacity, there will be 2 tables, one for the minimum flow that must be sent through the network and the other will be additional flow which can be found. Whereas if there is no minimum capacity on any edge, a single table will be displayed for all the flow sent through the graph. This can be seen in the magnification of section 3 in the diagram above. Each table will be identical with a column for path and another for flow with 2 buttons, (for up and down) to manipulate the scrollbar to view all the data. A total flow will be displayed below the scrollbar for all paths.

Section 4

For section 4, this displays the cut information in a similar format to tables in section 3, but with 4 columns instead, (the cut identifier with set notation, the path and values for the edges going from source to sink as well as sink to source and the total cut value) as can be seen in the magnification of this section in the image above. This allows the visibility for multiple cuts, grouping the associated information together. However, if there are no edges which are going from source to sink, a default message will be displayed identifying this.

Section 5

For section 5, this only displays text once the maximum flow is equal to the cut value drawn by the user, displaying an appropriate message about the theorem being proven, with the optimised flow found. This will be displayed over the top of section 1 once the graph has been solved.

Section 6

For section 6, messages will be displayed in 2 sections, separated by a border. One section is for comments aiding the solution of the problem and the other for errors, such as invalid cut or invalid textbox inputs. This will be in real time, running adjacent to the solution of the problem, allowing a greater understanding of the program and system.

Section 7

For section 7, this displayed the cut ID once the cut has been drawn and validated, enabling a hide button to run adjacent to it. This enables the user to hide certain cuts if they want to focus on a specific one.

1. Proposed solution:

A solution to the problem is to produce in python using pygame, a visual representation of any network flow problem, which is directly drawn based on the input parameters as the user is adding nodes and edges (Objective 7.1). Although an algorithm could be used to generate the coordinates for the graph, this will never coincide with the orientation of problems found in resources such as textbooks and past paper questions, often causing confusion and hindering learning due to the different positions of the node. Therefore, the user will enter the coordinates based on arbitrary positions on the axis where the graph will be drawn, in order to match the layout, while allowing forward planning for possible changes to the graph in further parts of the question, maintaining the correct orientation. This will also identify sources and sinks as the inputs are still being entered, adding in supersources and supersinks if needed, with the ability to hide the new nodes based of the user’s preference, (Objective 7.2). This will also enable the same ability for node capacities.

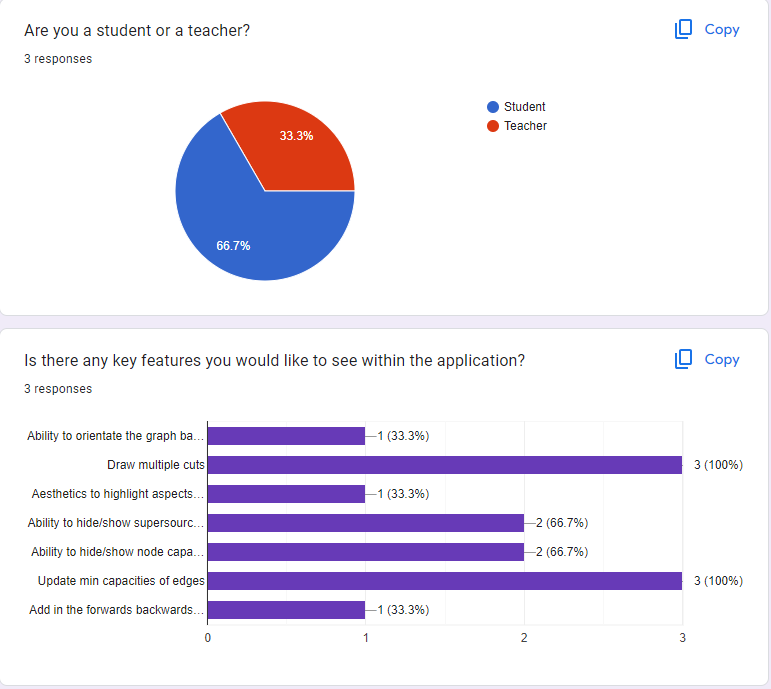
While inputs are being entered, the graph cannot be solved, thus once the ‘solve graph’ button has been pressed, the graph becomes more interactive. Flows can be entered by a user for a given path, presetting the forward flow before it has been solved. Once that ‘solve graph’ button has been pressed, the user will be able to access the maximum flow solution in segments rather than all at once (Objective 7.4.f), path by path, allowing the solution to run parallel to their explanation, which has the possibility to be split further into the minimum and maximum flow components. Once the maximum flow has been found, the path and flow automatically will be displayed as found in the solution. The program will allow a minimum cut to be drawn by the user, allowing multiple cuts to be represented, with notation automatically being displayed supporting the cut (Objective 7.5.a, 7.5.b, 7.5.c and 7.5.d). This can be hidden or shown based on the user inputs. Although a cut could be generated by an algorithm, the ability to make multiple cuts enables a greater variety of teaching points and potential patterns to be identified without sacrificing time in terms of making the cut. Through teaching aids and highlighting key features such as sources and sinks as well as patterns such as the backward flow being 0, users will better identify further paths or cuts as a result, (Objective 7.3.h, 7.4.e and 7.5.f). If a cut is found with the same value as the maximum flow, the program should evaluate proving the optimised flow for the given problem (Objective 7.f).

The proposed solution, due to it being a visual teaching tool, needs to have a simple GUI which clearly shows the problem, making it easily traversable and understandable, meaning a more custom GUI might be more suitable as this could be simplistic. This means for time efficiency, a more complicated GUI will require greater explanation and understanding of the application, hindering the useability.

1. Analysis performed:

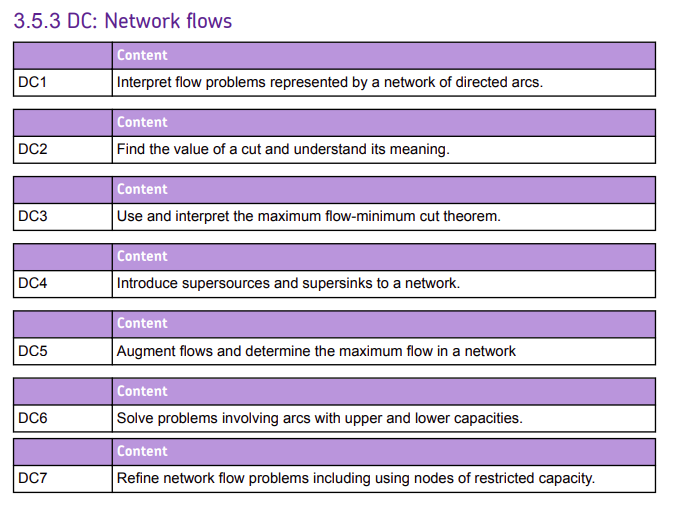
|  |  |  |
| --- | --- | --- |
| Date | Activity | Actions |
| 28th April 2023 | Informal chat with further maths department | Discussed the core aspect of interactivity regarding segmentation of solution, specifically during the maximum flow components.  Discussed the priorities of key features such as node capacities over general functionality such as saving. The specification for as and a level was also used for key features, which is displayed below under Specification |
| May/ June 2023 | Looked at past exam board questions as well as specification | Creates a list of objectives to include |
| 4th July 2023 | Informal chat with further maths department | Discussed potential limitations of the project and proposed solution to certain components such as minimum cuts.  Discussed potential changes in aesthetic to allow the user to see potential paths or find source and sink easier. |
| July 2023 | Chat with further maths department | Proposed the objectives and agreed |

My responses to a questionnaire that has been filled out



**A screenshot of a chat

Description automatically generated**

Appendix – Specification components: